Acheminement de messages instantanément stabilisant pour arbres couvrants †

Alain Cournier[1], Swan Dubois[2], Anissa Lamani[1], Franck Petit[3] et Vincent Villain[1]

[1] *MIS, Université de Picardie Jules Verne, France* [2] *Ecole Polytechnique Fédérale de Lausanne, Suisse* [3] *UPMC Sorbonne Universités & Inria, France*

Nous présentons un protocole instantanément stabilisant d'acheminement de messages au sein de structures couvrantes arborescentes de réseaux. Notre protocole utilise l'information fournie par un algorithme de calcul de tables de routage auto-stabilisant s'appuyant sur cette structure. Le fait que le protocole soit instantanément stabilisant signifie que tout message émis après les fautes est acheminé à son destinataire, y compris lorsque les tables de routage ne sont pas stabilisées. Notre algorithme présente l'avantage que le nombre de tampons est indépendant de tout paramètre global du réseau comme le nombre de nœuds ou le diamètre. En effet, nous montrons que le problème peut être résolu en utilisant un nombre constant de tampons par lien de communication de la structure couvrante. Cette propriété lui confère l'avantage de tolérer le passage à l'échelle.

Keywords: Acheminement de message, Stabilisation instantanée, Tolérance aux fautes

1 Introduction

Nous assistons ces dernières années à l'évolution de systèmes distribués à *large échelle*, par exemple les réseaux *pair-à-pair*. Leur taille et des considérations de performances imposent que ces systèmes s'appuient sur des structures couvrantes souvent utilisées comme des tables de hachage ou des index distribués. Nous nous concentrons sur les structures arborescentes. Parmi les grands défis à considérer dans le développement de tels systèmes se trouvent (*i*) le *passage à l'échelle*, (*ii*) la tolérance aux fautes et (*iii*) la prise en compte de la *dynamicité*, c'est-à-dire la propension du système à résister aux changements topologiques (ajout ou suppression de nœuds ou de liens). Les techniques utilisées pour tolérer les fautes sont habituellement classées en deux grandes familles. La première consiste à faire en sorte que les fautes soient *masquées* aux utilisateurs. Mais cette approche requiert souvent des mécanismes d'accord fondés sur de lourds échanges de messages du type "*question/réponse*" impliquant un grand nombre de nœuds. Ils s'avèrent assez peu adaptés aux très grands systèmes dynamiques. La seconde famille est l'*auto-stabilisation* qui garantit que si le système se trouve dans un état arbitraire, il retrouve de lui-même un comportement répondant à ces spécifications en un temps fini. L'auto-stabilisation se révèle donc être une approche naturelle pour développer des protocoles résistants aux fautes transitoires. En outre, l'auto-stabilisation se montre également être une bonne technique pour tolérer les passages à l'échelle et la dynamicité du système, en particulier lorsque les protocoles ne requièrent pas la connaissance d'un ou plusieurs paramètres globaux du système comme le nombre de nœuds ou le diamètre du réseau.

Nous nous intéressons au problème de la *communication point-à-point* dans les systèmes à large échelle. Certains nœuds du système (appelés nœuds émetteurs) souhaitent envoyer des messages à d'autres (appelés nœuds destinataires). Le problème consiste à délivrer tous ces messages en un temps fini. Ce problème englobe en réalité deux sous problèmes : (*i*) le problème du routage —calcul du chemin à suivre par les messages de l'émetteur vers le destinataire— et (*ii*) le problème d'acheminement de messages —gestion des

†Ce travail a été financé par le projet ANR SPADES (08-ANR-SEGI-025, `http://graal.ens-lyon.fr/SPADES`). Une version plus longue a été présentée à ICDCN 2012, Hong Kong. La version détaillée est disponible à l'adresse suivante : `http://hal.inria.fr/inria-00608897`.

ressources allouées au transport de messages. Le routage est fortement relié au problème de la construction d'arbres couvrants.

Il existe de nombreuses solutions auto-stabilisantes permettant de résoudre le problème du routage [6]. Des solutions auto-stabilisantes ont été proposées pour résoudre le problème d'acheminement de message dans [1, 7], mais elles ont l'inconvénient de ne pas pouvoir éviter l'éventuelle perte de messages, même après l'arrêt des fautes. La *stabilisation instantanée* [2] est une branche de l'auto-stabilisation. Elle garantit que le système, indépendamment de son état initial, a *toujours* un comportement répondant à ses spécifications. Dans le contexte qui nous intéresse, ceci veut dire qu'un protocole d'acheminement de messages instantanément stabilisant associée à un algorithme de routage auto-stabilisant garantit que tout message émis après une faute transitoire ou un changement topologique est délivré à sa destination en un temps fini. Ceci contraste avec une solution qui n'est que "simplement" auto-stabilisante (sans être instantanément stabilisante) car elle garantit uniquement que le nombre de messages non délivrés est borné.

Les premiers algorithmes d'acheminement instantanément stabilisant fournis par la littérature [4, 5] nécessitent un nombre de tampons par nœud proportionnel respectivement au nombre de nœuds et au diamètre du réseau, ce qui ne permet pas le passage à l'échelle du réseau. Pour remédier à cela, dans [3], nous avons proposé une solution ne nécessitant qu'un nombre de tampons par nœud proportionnel au degré du nœud. Malheureusement, cette solution n'est pas satisfaisante puisque qu'elle nécessite que la structure couvrante soit réduite à (ou traitée comme) une chaîne.

Dans cet article, nous fournissons un protocole d'acheminement instantanément stabilisant nécessitant un nombre constant de tampons par lien de communication de la structure couvrante. Cette propriété lui confère l'avantage de tolérer le passage à l'échelle. Dans ce qui suit, nous nous attachons à présenter notre solution de manière synthétique.

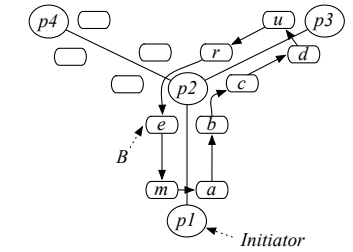
2 Présentation synthétique

Nous supposons l'existence d'un algorithme de calcul des tables de routage auto-stabilisant s'exécutant en parallèle de notre algorithme. Nous supposons que tout nœud p a accès aux tables de routage qui retournent pour toute destination d l'identité du voisin par lequel un message de destination d doit transiter.

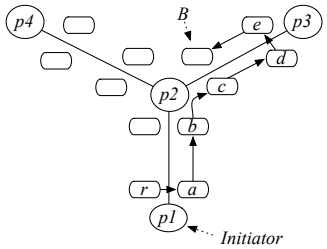
Notons $\delta(p)$ le degré d'un nœud p. Chaque nœud p a (i) un tampon interne, (ii) $\delta(p)$ tampons d'entrée lui permettant de recevoir des messages de ses voisins (pour chaque voisin q de p, le tampon d'entrée de p connecté au lien (p,q) est noté $IN_p(q)$), et (iii) $\delta(p)$ tampons de sortie permettant au nœud p d'envoyer des messages à ses voisins (pour $q \in N_p$, le tampon de sortie de p connecté au lien (p,q) est noté $OUT_p(q)$). En d'autres termes, chaque nœud p possède $2\delta(p)+1$ tampons. La génération d'un nouveau message se fait toujours dans le tampon de sortie du lien (p,q) où q est le voisin indiqué par la table de routage par lequel doit transiter le message afin de parvenir à sa destination.

L'idée de l'algorithme est la suivante : quand un nœud veut générer un nouveau message, il consulte sa table de routage pour déterminer par quel voisin doit transiter le message pour atteindre sa destination. Une fois ce message généré, il est routé selon les tables de routage : notons $nb(m,b)$ le tampon dans lequel doit être transmis un message m actuellement stocké dans le tampon b. Nous avons alors les deux propriétés suivantes : (i) pour tout message m, $nb(m,IN_p(q)) = OUT_p(q')$ où q' est le prochain nœud par lequel m doit transiter pour atteindre sa destination (d'après la table de routage) et (ii) pour tout message m, $nb(m,OUT_p(q)) = IN_q(p)$. En d'autres termes, (i) signifie que si un message m est dans un tampon d'entrée de p ($IN_p(q)$) et que p n'est pas la destination de m, p consulte ses tables de routage pour déterminer le prochain nœud q' par lequel le message doit transiter. Le message est alors copié dans le tampon de sortie de p le reliant à q ($OUT_p(q)$). D'autre part, (ii) signifie que, si un message m est dans un tampon de sortie de p ($OUT_p(q)$) et que p n'est pas la destination de m, il sera copié dans le tampon d'entrée de q ($IN_q(p)$). Notez que lorsque les tables de routage sont correctes et que tous les messages sont dans la "bonne direction" (c'est-à-dire correctement routés vers leur destinataire), (i) n'est jamais vérifié pour $q = q'$. Néanmoins, cela peut être le cas lorsque les tables de routage sont en cours de stabilisation et s'il y a des messages routés dans la mauvaise direction, c'est-à-dire dirigé dans un sous arbre qui ne contient pas la destination.

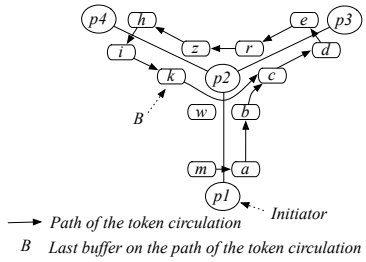
Avant de continuer l'explication de notre solution, revenons d'abord sur la progression de message dans les tampons. Un tampon est dit *libre* si et seulement s'il ne contient aucun message ou s'il contient le même



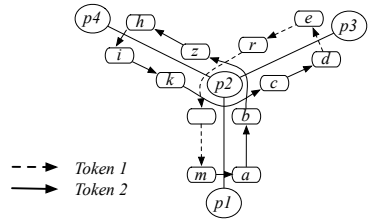
(a) Exemple de circuit complet



(b) Tampon libre sur un chemin



(c) Example de sous-circuit



(d) Exemple d'interblocage entre circulations de jetons

FIGURE 1: Quatre exemples associés à la circulation de jeton.

message que le tampon d'entrée auquel il est relié. Par exemple, si $IN_p(q) = OUT_q(p)$ alors $OUT_q(p)$ est dit *libre*. Dans le cas contraire, le tampon est dit *occupé*. La progression des messages implique le remplissage et la libération de tampons, c'est-à-dire que chaque tampon est alternativement *libre* et *occupé*. Ce mécanisme induit clairement que des emplacements *libres* se déplacent dans le graphe formé par les tampons, un emplacement libre correspondant à un tampon libre à un instant donné.

Dans la suite, nous appelons *arc actif* un arc entre deux tampons dont le tampon source est *occupé*. Dans ce cas, si le sous-graphe constitué par les arcs actifs comporte des circuits alors nous sommes en présence d'une situation d'interblocage. L'état initial étant arbitraire et les tables de routage stabilisant en un temps fini, ce type de circuit ne peuvent apparaître que sur un préfixe fini de l'exécution (voir Figure 1, (a)).

Étant donné la topologie arborescente de notre système, l'existence d'un circuit implique l'existence de deux messages m et m' tels que : $nb(m, IN_p(q)) = OUT_p(q)$ et $nb(m', IN_{p'}(q')) = OUT_{p'}(q')$ (voir Figure 1, (a)). La localité de l'information implique qu'il est impossible pour un nœud p de connaitre l'existence d'un circuit. Il peut juste suspecter la présence d'un cycle dans le cas où un message m dans son tampon d'entrée $IN_p(q)$ doit être copié dans $OUT_p(q)$. Pour s'en assurer, p initie un mécanisme de requête fondé sur une circulation de jeton. Il suit les arcs actifs en commençant par le tampon qui contient le message m. De cette manière, p détecte en un temps fini un tampon vide (voir Figure 1) ou un circuit. Notez que deux types de circuits existent : (i) le *circuit complet* impliquant le premier tampon contenant le message m visité par le jeton (voir Figure 1, (a)) ou (ii) le *sous-circuit* n'impliquant pas ce tampon (voir Figure 1, (c)).

Lorsque la circulation de jeton détecte un tampon vide (désignons ce tampon par B), l'idée est de faire avancer les messages sur le chemin de la circulation de jeton pour permettre à l'emplacement vide initialement sur B d'avancer. De la sorte, nous nous assurons que $OUT_p(q)$ sera *libre* en un temps fini ce qui permettra à p de copier le message m directement dans $OUT_p(q)$ (cette action ayant la priorité sur toutes les autres actions touchant $OUT_p(q)$). Si la circulation de jeton détecte un circuit, deux sous-cas sont possibles : (i) le circuit détecté est complet et (ii) le circuit détecté est un sous-circuit. Dans le cas (i), le but sera de libérer le tampon $OUT_p(q)$ si le nœud p est celui qui a détecté le cycle (p_1 dans la Figure 1, (a)). Dans le cas (ii), le nœud ayant le dernier tampon B visité par la circulation de jeton est celui qui détecte le cycle (nœud p_2 dans la Figure 1, (c)). Notez que B est dans ce cas un tampon d'entrée. Le but est de libérer le tampon de sortie B' par lequel le message présent dans B doit transiter afin d'atteindre sa destination ($OUT_{p_2}(p_3)$ dans la Figure 1, (c)). Notez que B' a été visité par la circulation de jeton. Dans les deux cas, le nœud qui détecte

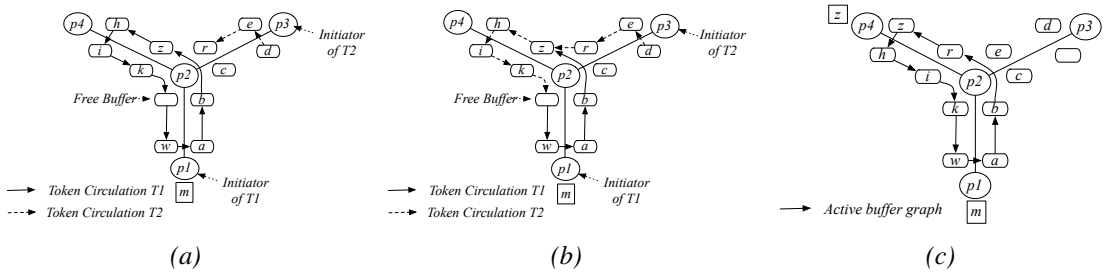


FIGURE 2: Trois exemples de problèmes.

le cycle copie le message du tampon d'entrée correspondant ($IN_p(q)$ ou B) dans son tampon interne. En faisant cela, le nœud libère son tampon d'entrée. L'idée est alors de faire avancer les messages tout au long du chemin de la circulation de jeton pour faire avancer l'emplacement vide qui a été créé. Cela assure que le tampon de sortie visé sera libre en un temps fini ($OUT_p(q)$ ou B'). Le message dans le tampon interne peut alors être copié dans l'emplacement vide se trouvant dans le tampon de sortie.

Plusieurs circulations de jeton peuvent être initiées et exécutées en parallèle. Afin d'éviter des situations d'interblocage entre elles (voir Figure 1, (d)), chaque circulation de jeton comporte l'identifiant de son nœud initiateur. Une circulation de jeton ayant l'identifiant id sera autorisée à passer par un tampon déjà visité par une autre circulation de jeton ayant l'identifiant id' si $id < id'$. Notez que cela peut engendrer la perte d'emplacement vide récupéré par une circulation de jeton donnée. A titre d'exemple, dans la Figure 2, on peut observer que l'emplacement vide généré par $T1$ est pris par $T2$. En faisant avancer les message sur le chemin de $T2$, un nouveau circuit est créé impliquant p_1 et p_4. Si on suppose que cette situation se produit à nouveau de sorte que le tampon interne de p_4 devient *occupé* et que p_4 et p_1 soient simultanément dans un autre circuit, le système est alors interbloqué. Notez que ce circuit ne peut être supprimé étant donné qu'aucun message valide ne peut être supprimé. Nous devons donc éviter qu'une telle configuration survienne. Pour cela, lorsqu'une circulation de jeton détecte un tampon vide ou un circuit, elle parcours le chemin inverse afin de valider celui-ci, interdisant ainsi à toute autre circulation de jeton d'utiliser ses tampons. Notez que le jeton revient à l'initiateur de la circulation de jeton. L'initiateur renvoie encore le jeton afin de vérifier que tout le chemin de la circulation de jeton a été bien validé (*i.e.* que la circulation de jeton n'a pas fusionné avec une autre qui était présente dans la configuration initiale). Nous dirons alors que le chemin a été confirmé. Une fois le chemin confirmé, l'emplacement vide peut être créé et déplacé afin de libérer le tampon de sortie du nœud visé.

Références

[1] B. Awerbuch, B. Patt-Shamir, and G. Varghese. Self-stabilizing end-to-end communication. *Journal of High Speed Networks*, 5(4) :365–381, 1996.

[2] A. Bui, A. Datta, F. Petit, and V. Villain. Snap-stabilization and pif in tree networks. *DC*, 20(1) :3–19, 2007.

[3] A. Cournier, S. Dubois, A. Lamani, F. Petit, and V. Villain. Snap-stabilizing linear message forwarding. In *SSS*, pages 546–559, 2010.

[4] A. Cournier, S. Dubois, and V. Villain. A snap-stabilizing point-to-point communication protocol in message-switched networks. In *IPDPS*, pages 1–11, 2009.

[5] A. Cournier, S. Dubois, and V. Villain. How to improve snap-stabilizing point-to-point communication space complexity ? *TCS*, 412(33) :4285–4296, 2011.

[6] Shlomi Dolev. Self-stabilizing routing and related protocols. *Journal of Parallel and Distributed Computing*, 42(2) :122 – 127, 1997.

[7] E. Kushilevitz, R. Ostrovsky, and A. Rosén. Log-space polynomial end-to-end communication. In *STOC '95 : Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 559–568. ACM, 1995.